REMARKS

Claims 22 to 44 were pending when last examined. Applicant has amended claims 22, 23, 35, 37, and 41.

The Claimed Invention

As described in the specification and the numerous references lauding the present invention, the major benefit of the claimed invention is that the customer does not need to program their embedded processor to handle the GUI (i.e., the CPU draws the GUI and interacts with the user, or the CPU instructs a graphic controller how to draw the GUI and interact with the user). See Specification, p. 17, lines 15 to 24. Instead, the GUI controller is provided to handle the GUI totally independent of the embedded processor. See Specification, p. 5, line 12 to p. 6, line 8; p. 24, lines 13 to 23. The GUI controller is easily programmed with a HTML document that defines the GUI with the GUI objects in the GUI library. See Specification, p. 17, line 24 to p. 18, lin3. The HTML document is compiled and then stored in the GUI controller for execution. The customer can readily change the HTML document to modify the GUI.

§ 103(a) Rejections

The Examiner rejected claims 22 to 44 under 35 U.S.C. § 103(a) as being unpatentable over U.S. Patent Nos. 5,995,120 ("Dye") and 6,118,462 ("Margulis").

Addressing Applicant's arguments from the January 5, 2005 Response to Office Action, the Examiner "noted that the features upon which applicant relies (i.e., operations completely independent of the CPU) are not recited in the rejected claim(s)." April 21, 2005 Final Office Action, p. 10. Furthermore, the Examiner found that Dye discloses supplemental libraries that "help in the rendering of the objects appearances and functions" and the claims "only require[] sets of executable codes for rendering the GUI object." Id.

In response, Applicant has amended independent claims 22 as follows:

> 22. A control system for a device, wherein a graphic user interface ("GUI") controller operates a GUI of the device independently from an embedded controller of the device, the control system comprising:
>
> the embedded controller for controlling and monitoring the device;

a liquid crystal display ("LCD") for displaying the GUI to a user, the GUI including:

a first GUI object for displaying a status of the device; and

a second GUI object for displaying a command to the device;

a touch screen for detecting the command from the user;

the GUI controller, comprising:

at least one memory, comprising:

a document buffer storing a document defining an appearance of the GUI, the document comprising:

a first operation code ("opcode") identifying the first GUI object;

a second opcode identifying the source of the status;

a third opcode identifying the second GUI object; and

a fourth opcode identifying the destination of the command;

a data buffer storing the status and the command;

a GUI object library storing:

a first set of executable codes defining an appearance and a functionality of the first GUI object with instructions for rendering the first GUI object, receiving the status from the embedded processor, and further rendering the first GUI object to show a visual response to the status;

a second set of executable codes defining an appearance and a functionality of the second GUI object with instructions for rendering the second GUI object, receiving the command from the touch screen, further rendering the second GUI object to show a visual response to the command, and sending the command to the embedded processor;

a frame buffer storing at least one complete display frame image of the GUI;

a GUI processor for rendering the GUI and handling user inputs independently from the embedded controller, wherein:

the GUI processor is coupled to the touch screen via a touch screen analog to digital converter to receive the command;

the GUI processor is coupled to the embedded processor via a serial UART interface to send the command and to receive the status;

the GUI processor is coupled to the at least one memory via a direct memory bus interface, wherein:

in response to the first and the second opcodes, the GUI processor executes the first set of executable codes to render the first GUI object to the frame buffer independently from the embedded processor, to communicate with the embedded processor to receive the status from the embedded processor, and to further render the first GUI object to the frame buffer in response to the status independently from the embedded processor;

in response to the third and the fourth opcodes, the GUI processor executes the second set of executable codes to render the second GUI object to the frame buffer independently from the embedded processor, to receive the command from the touch screen independently from the embedded processor, to render the second GUI object again to the frame buffer to show a visual response to the command independently from the embedded processor, and to send the command to the embedded processor;

a pixel serializer coupled to the LCD to continuously refresh the LCD with the complete display frame image in the frame buffer that contains both the rendered first GUI object and the rendered second GUI object.

Claim 22 (emphasis added).

Applicant submits that Dye does not disclose a GUI controller with a GUI processor that operates a GUI of a device independently from an embedded controller of the device. As discussed in the January 5, 2005 Response to Office Action, while the integrated memory controller (IMC) of Dye improves the screen update process by the use of a display refresh list, the IMC does not

operate independently from the CPU to draw the windows/objects and to interact with the user. Per the Examiner's request in the telephone interview on July 20, 2005, Applicant provides the following paragraphs of Dye in support of Applicant's argument.

> Video screen changes or screen updates are preferably performed using the following operations. First, in response to software executing on the host CPU, such as applications software, the video driver executing on the CPU generates a video driver instruction list which includes screen update and/or graphics information for displaying video data on the screen. The video driver instruction list is provided to the Execution Engine in the graphics controller or IMC. The Execution Engine examines the video driver instruction list and generates a list of graphics and/or memory commands to the Graphics Engine. Thus the Execution Engine constructs a complete list of graphics or memory operations to be performed in response to desired screen change information.

Dye, col. 3, line 62 to col. 4, line 7.

> The CPU 102 begins program execution by reading the recently decompressed program code from the system memory 110. Portions of the program code contain information necessary to write data and/or instructions back to the IMC 140 using a special graphical protocol according to the present invention to direct the IMC 140 to control the display output on the video display 142. In many cases, the graphical data is not required to leave the system memory 110 and is not required to move to another location in system memory 110, but rather the display list-based operation and high level graphical protocol of the IMC 140 of the present invention enables the CPU 102 to instruct the IMC 104 how window and other graphical data is presented on the screen. This provides a tremendous improvement over prior art systems.
>
> .... Instead, a computer system incorporating an IMC 140 according to the present invention includes a high level graphic protocol whereby the CPU 102 instructs the IMC 140 to manipulate the data stored in the system memory 110. For example, when text which appears in a window on the display screen is manipulated, the text is not required to leave the system memory 110 for processing by the CPU 102. Rather, the IMC 140 reads the text data into the system memory 110, preferably in ASCII format, and the IMC 140 processes the text data for display output. This operation is performed under the direction of the CPU 102 through the high level graphic protocol used by the IMC 140, as described further below. ....

Dye, col. 11, lines 22 to 53 (emphasis added).

> A video driver executes on the CPU 102 and generates a video driver instruction list which includes video display information regarding desired changes to the video output of the video monitor 142. The video display information includes screen update and/or graphics information for displaying video data on the display screen of the video monitor 142. For example, the video

> display information may include commands to draw a 3D texture map, or to bit blit pixel data from a first xy memory location to a second xy memory location, to render a polygon, or to assemble a display refresh list.
>
> The video driver instruction list is provided to the Execution Engine 210 in the graphics controller or IMC. The Execution Engine 210 examines the video driver instruction list and generates a list of graphics and/or memory commands to the Graphics Engine 212. Thus the Execution Engine 210 constructs a complete list of graphics or memory operations to be performed in response to desired screen change information. In response to an Assemble Display Refresh List command, the Execution Engine 210 assembles or constructs a display refresh list.

Dye, col. 21, lines 6 to 26 (emphasis added). As can be seen from above, the CPU instructs the IMC on how to display windows and other graphical objects and the IMC generates a display refresh list accordingly. Thus, the IMC does not operate a GUI independently from the CPU as recited in claim 1.

Applicant further submits that Dye does not disclose a GUI object library that defines the appearances and the functionalities of GUI objects. As discussed in the January 5, 2005 Response to Office Action, Dye appears to disclose that software drivers can call supplemental libraries to assist in the rendering of the windows/objects. While these supplemental libraries assist in the rendering the appearances and the functions of the windows/objects, these supplemental libraries do not actually define the appearance and the functionality of the GUI objects as recited in claim 1.

Margulis does not cure the deficiencies of Dye as recited against claim 22. Thus, claim 22 is now patentable over the combination of Dye and Margulis.

Applicant has amended claim 23 with similar limitations as claim 22. Thus, claim 23 is patentable over Dye and Margulis for at least the same reasons as claim 22.

Claims 24 to 34 depend from claim 23 and are patentable over the cited references for at least the same reasons as claim 23.

Applicant has amended claim 35 with similar limitations as claim 22. Thus, claim 35 is patentable over Dye and Margulis for at least the same reasons as claim 22.

Claims 36 to 44 depend from claim 35 and are patentable over the cited references for at least the same reasons as claim 35.

In summary, claims 22 to 44 were pending when last examined. Applicant has amended claims 22, 23, 35, 37, and 41. Applicant respectfully requests the allowance of claims 22 to 44. Should the Examiner have any questions, the Examiner is invited to call the undersigned at (408) 382-0480.

Certification of Facsimile Transmission

I hereby certify that this paper is being facsimile transmitted to the U.S. Patent and Trademark Office on the date shown below.

Signature 7/21/2005 Date

Respectfully submitted,

David C. Hsia
Attorney for Applicant(s)
Reg. No. 46,235